Facultad de Estudios Estadísticos. Grado en Estadística Aplicada. Curso 2013-2014.

Agenda semanal de tareas y reuniones

Esta práctica consiste en desarrollar un programa que permita gestionar las tareas y reuniones de una persona, al estilo de aplicaciones como *Google Calendar*. El objetivo del programa es almacenar las tareas y las reuniones que debe realizar el usuario, así como encontrar el mejor momento en el que se puede planificar una reunión o una tarea sin coincidir con otras tareas.

1. Tipos de datos.

- Una tarea es una actividad a realizar a una hora y con una duración determinadas. Se va a utilizar el mismo tipo de datos para representar tanto las tareas individuales como las reuniones. Una tarea se debe representar mediante un tipo de datos **`tTarea`** con la siguiente información:
  - Identificador: una cadena de hasta 5 caracteres que permita identificar la tarea.
  - Descripción de la tarea.
  - Día de la semana en el que se realiza (entero de 0 a 4 –lunes a viernes; -1 si se realiza todos los días).
  - Hora de inicio, un número de 0 a 10 que representa la hora (0 representa las 8:00, 1 las 9:00, etc. Todas las tareas empiezan a las horas en punto).
  - Duración de la tarea (en horas), un número entero positivo.
  - Número de participantes en la reunión (0 si es una tarea individual).
  - Una lista con los nombres de los participantes. Se permitirán hasta P participantes en cada reunión.
- Para representar la información de la agenda, se debe crear un tipo de datos **`tAgenda`** con la siguiente información:
  - Número de tareas.
  - Una lista con las tareas a realizar durante la semana. Cada tarea se representa mediante el tipo de datos **`tTarea`** detallado anteriormente. Se pueden tener hasta K tareas.

2. Funcionalidades que se deben programar.

- Escribe un procedimiento **`menu()`**, llamado desde **`main()`**, que debe crear una variable de tipo **`tAgenda`**, inicializarla y a continuación mostrar en pantalla el menú principal del programa. Las opciones del menú son las siguientes:

```
1. Leer la agenda de un fichero en disco.
2. Guardar la agenda en un fichero en disco.
3. Añadir una tarea a la agenda.
4. Mostrar en pantalla la lista de reuniones y tareas.
5. Mostrar en pantalla la vista semanal de la agenda.
6. Recolocar automáticamente una tarea.
q. Salir del programa.
```

  La inicialización de la variable de tipo **`tAgenda`** debe hacerse en un subprograma con el siguente prototipo: `void inicializar(tAgenda& ag)`.

  Cuando se teclee una opción, el procedimiento **`menu()`** debe realizar las funciones correspondientes a esa opción y después deberá volver a mostrar el menú en pantalla para que el usuario seleccione la siguiente opción. El programa debe terminar cuando se pulse la opción "q".

  Crea el procedimiento **`menu()`** de forma que llame a subprogramas distintos para cada una de las opciones del menú. Estos subprogramas deben estar inicialmente vacíos, pues los iremos rellenando a medida que avanzamos en su desarrollo.

Compila el programa y prueba que el procedimiento `menu()` funciona correctamente mostrando el menú cada vez que se teclea una opción. El programa debe terminar cuando se teclea 'q'.

- Escribe el código del subprograma correspondiente a la opción 3. Para implementar esta opción, utiliza un subprograma al que se pase como parámetro por referencia la variable de tipo **`tAgenda`** declarada en **`menu()`** para poder modificar su valor.

  Este subprograma debe comprobar primero si hay espacio para añadir una tarea más en la lista de tareas (si ya hay K tareas en la agenda, no se pueden añadir más). Si no hay espacio suficiente, se debe escribir un mensaje de error en la pantalla y volver al menú principal.

  En caso contrario, este subprograma debe leer del teclado los datos de una tarea nueva y almacenarlos en un elemento de la estructura de tipo **`tAgenda`** declarada en **`menu()`**, actualizando adecuadamente el número de tareas. Debe comprobarse que los datos introducidos son correctos: el identificador debe tener como máximo 5 caracteres, el día de la semana debe estar entre -1 y 6, la hora de inicio entre 0 y 10, y la duración de la tarea debe ser un número positivo entre 1 y 24 menos la hora de inicio (es decir, solo se permiten tareas hasta la medianoche). El número de participantes debe estar entre cero y P. Si es mayor a cero, debe leer el nombre de los participantes. También se debe comprobar que el identificador de la tarea no se utiliza en ninguna otra tarea de la lista de tareas de la agenda. Si algún dato es incorrecto, debe mostrar un mensaje de error y volver a solicitar el dato.

  Una vez introducidos los datos de la tarea nueva, se debe comprobar si la tarea se solapa con alguna otra tarea. Si se solapa, se debe mostrar un mensaje indicando la(s) tarea(s) con las que se solapa. A continuación se solicitará confirmación de que el horario es el correcto. En caso afirmativo se añadirá la tarea, aunque se solape con otras.

  IMPORTANTE: Utiliza subprogramas para leer y comprobar cada uno de los datos leídos del teclado, además de otros auxiliares que puedas necesitar:

```
tTarea leerDatosTarea(); //esta debe llamar a las siguientes:
string leerIdentificador();  //el tipo de la funcion puede ser distinto.
bool buscarTarea(tAgenda ag, ...); //completa los parámetros.
int leerDiaSemana();
int leerHora();
...
bool haySolapamiento(tTarea t, tAgenda ag);
void aniadirTarea(tAgenda& ag, tTarea t);
```

  Antes de continuar, compila el programa y prueba que esta opción del menú funciona correctamente.

- Escribe el código del subprograma correspondiente a la opción 4: Debe mostrar en pantalla los datos de las tareas y reuniones almacenadas en la memoria, **ordenadas por día y hora de inicio**. Para ello, el subprograma debe tener como parámetro **por referencia** la variable de tipo **`tAgenda`** declarada en **`menu()`**. Las tareas pueden no estar ordenadas cronológicamente. Antes de mostrarlas se deben ordenar utilizando el algoritmo de ordenación visto en clase.

  Un ejemplo de lista de tareas (algunas son reuniones) es el siguiente:

```
LISTA DE TAREAS:
------------------------------------------------------
DP1  : *** 09:00-14:00 - Direccion de proyectos.
SGWEB: Lun 16:00-18:00 - Seguimiento servicio Web.
           Reunion con:  Arturo Garcia Rivero
                         Teresa Gonzalez Gonzalez-Romero
                         Evaristo Garcia Gomez
ADM1 : Mie 16:00-18:00 - Administracion.
PERS1: Jue 08:00-09:00 - Reconocimiento medico.
```

```
MKTNG: Jue 16:00-19:00 - Planificacion Marketing.
           Reunion con:  Teresa Gonzalez Gonzalez-Romero
                         Manuel Lopez Fernandez
BBDD : Jue 17:00-18:00 - Seguimiento BBDD.
ADM2 : Vie 17:00-19:00 - Administracion.
```

Al final de la lista se debe mostrar el siguiente mensaje:

```
-----------------------------------------------------------
Introduzca una opción: 'm <id>' para modificar una tarea,
                       'e <id>' para eliminar una tarea,
                       'q' para volver al menú principal
```

Si el usuario selecciona la opción **`m <id>`** debe mostrar todos los datos de la tarea cuyo identificador es **`<id>`** y permitir introducirlos de nuevo por el teclado. Por ejemplo, si se teclea **`m MKTNG`**, debe mostrar sus datos:

```
-----------------------------------------------------------
MKTNG: Jue 16:00-19:00 - Planificacion Marketing.
           Reunion con:  Teresa Gonzalez Gonzalez-Romero
                         Manuel Lopez Fernandez
Teclee los nuevos datos de la tarea MKTNG:
Descripcion: ...
```

Para leer de nuevo los datos de la tarea, llama a los subprogramas de lectura de datos que has definido en el punto anterior.

Si el usuario selecciona la opción **`e <id>`**, debe mostrar todos los datos de la tarea y pedir confirmar la eliminación. Cuando se elimina una tarea de la agenda, se debe dejar la lista de tareas en un estado consistente: se deben mover todas las tareas posteriores de la lista una posición adelante, para no dejar huecos en la lista de tareas.

IMPORTANTE: Utiliza subprogramas para cada una de las opciones `m` y `e`, además de otros auxiliares que puedas necesitar:

```
void mostrarDatosTarea(tTarea t);
void modificarDatosTarea(tTarea& p);
void eliminarTarea(tAgenda& ag, ...); //completa los parámetros.
...
```

Antes de continuar, compila de nuevo el programa y prueba que esta opción del menú funciona correctamente.

- Escribe el código de los subprogramas correspondientes a las opciones 2 y 1, en este orden. Para ello, debe leer del teclado el nombre del archivo en el que se va a guardar/leer los datos de la agenda. A continuación, se debe llamar a otro subprograma que realice la lectura/escritura de la agenda.

  Recuerda que el archivo se debe guardar con la opción 2 en un formato tal que se pueda recuperar con la opción 1.

  Comienza con la opción 2. Compila el programa y prueba que se escribe correctamente el archivo, abriéndolo con el bloc de notas. Después escribe el código correspondiente a la opción 1. Verifica que los datos que se han escrito en el archivo se leen después correctamente.

- Escribe el código del subprograma correspondiente a la opción 5. Debe mostrar las tareas y reuniones de acuerdo con el formato del siguiente ejemplo:

```
============================================================
HORA  | LUNES     | MARTES    | MIERC.    | JUEVES    | VIERNES  |
============================================================
08:00 |           |           |           | PERS1     |          |
09:00 | DP1       | DP1       | DP1       | DP1       | DP1      |
10:00 | DP1       | DP1       | DP1       | DP1       | DP1      |
11:00 | DP1       | DP1       | DP1       | DP1       | DP1      |
12:00 | DP1       | DP1       | DP1       | DP1       | DP1      |
13:00 | DP1       | DP1       | DP1       | DP1       | DP1      |
14:00 |           |           |           |           |          |
15:00 |           |           |           |           |          |
16:00 | SGWEB (R) |           | ADM1      | MKTNG (R) |          |
17:00 | SGWEB (R) |           | ADM1      | MKTNG*(R) | ADM2     |
18:00 |           |           |           | MKTNG (R) | ADM2     |
============================================================
```

  Se recomienda hacer una matriz en memoria que refleje cada una de las horas de cada día de la semana, que se irá rellenando con el identificador de tarea o reunión correspondiente. Si en una hora se solapan dos tareas o reuniones, se debe marcar la primera de las tareas con un asterisco. En el ejemplo anterior existe otra tarea el jueves a las 17:00.

  Si una tarea es una reunión con otros participantes, debe indicarse con **`(R)`** (de "reunión") en cada casilla en la que aparece.

- Escribe un subprograma correspondiente a la opción 6 del menú. Debe pedir que se teclee el identificador de una tarea. Se debe comprobar que la tarea exista en la agenda y que no esté planificada para todos los días (el día debe ser mayor o igual a cero).

  A continuación debe buscar un día y hora de inicio de forma que no se solape con ninguna otra tarea. Se recomienda utilizar el subprograma `haySolapamiento()` definido anteriormente para comprobar cuándo no hay solapamiento para encontrar el día y hora al que recolocarla.

  Si se ha encontrado una hora en la que no se solapa, se debe mostrar en la pantalla y pedir confirmación al usuario. En caso afirmativo, se procederá a recolocarla cambiando el día y hora de inicio. Si no hay ningun día y hora para recolocarla, debe mostrar un mensaje indicándolo y volver al menú principal.